Validation of Requirements for Embedded Software using Petri Nets

Prerna Kanojia

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769008, Odisha, India

Validation of Requirements for Embedded Software using Petri Nets

Thesis submitted in partial fulfilment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Software Engineering)

by

Prerna Kanojia

(Roll No. 212CS3121)

under the supervision of

Prof. S. K. Rath

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769008, India

June 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***Validation of Requirements for Embedded Software using Petri Nets*** by ***Prerna Kanojia*** is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology with the specialization of Software Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: June 1, 2014

Dr. S. K. Rath
Professor, CSE Department
NIT Rourkela, Odisha

Acknowledgment

I am thankful to various local and global peers who have contributed towards shaping this thesis. At the outset, I Would Like To Express my sincere gratitude towards Professor S. K. Rath for his invaluable guidance and continuous support throughout my graduate studies. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge.

I am very grateful to all my friends who, in a way or another, have contributed to making this thesis possible. My colleagues at the Advanced Software engineering Laboratory (ASE Lab) Sumana, Amar, Jyoti, Lov, and lab seniors Y. Suresh, Abinash Tripathy, Mukesh Kumar, Shashank M. Satapathy and Ashish Dwivedi who have created a friendly working environment. Thank you all.

My sincere thanks to my friend Sigampalli Sudheer who has provided me with kind words, a welcome ear, new ideas, useful criticism, and his invaluable time, I am truly indebted.

I would also like to thank specially Avanish Kumar (Best Friend) for standing besides me all the time and support me morally and ethically.

I must acknowledge the academic resources that I have got from NIT Rourkela. I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles.

Last, but not the least, I would like to dedicate this thesis to my parents, sister and brother for their love, patience, and understanding.

Prerna Kanojia
Roll: 212CS3121

Abstract

Embedded systems are utilized as a part of a wide range of spectrum extending from home apparatuses and cell phones to medical apparatus and transport controllers. They are commonly portrayed by their real-time behavior and must satisfy strict requirements on reliability and accuracy. The key challenge in real time system analysis is that proper scheduling strategy needs to be assured. So, the validation of requirements for these systems must be assured. Petri net is a formal and executable modeling technique, most suitably used for analysis of any concurrent system. There are a number of tools based on Petri net theory for analysis of models that help the users to graphically analyze a model, simulate them through an animated sequence and use them to validate the process.

In this thesis, the proposed approach makes an attempt to model and analyze a case study on real time system i.e., Elevator Control System. First, the system is modeled using Colored Petri nets which can acquire the essential properties of the system and allow its illustration at different level of granularity. Second, after modeling the system, performance analysis is carried out using Stochastic Petri nets to analyze various aspects of the system. Third, verification and validation of the model is conducted using TAPAAL tool of Petri nets to check the correctness and to prove whether certain properties, demonstrated as computational tree logic formulas, hold concerning the system model. The use of three Petri net tools for analysis of any real time system helps to validate the workflow and subsequently, proper design of software architecture.

Keywords: *Colored Petri nets, Computational Tree Logic, Petri nets, Stochastic Petri nets.*

Contents

List of Figures

List of Tables

List of Abbriviations

PN	Petri nets
CPN	Colored Petri nets
HCPN	Hierarchical Colored Petri nets
SPN	Stochastic Petri nets
GSPN	Generalized Stochastic Petri nets
PIPE	Platform Independent Petri net Editor
CTL	Computation Tree Logic
MC	Markov Chain
TAPN	Timed Arc Petri nets
TPN	Timed Petri nets
PNML	Petri net Markup Language

Chapter 1

Introduction

Motivation

Objective

Organization of Thesis

Chapter 1

Introduction

1.1 Introduction

Embedded systems are utilized as a part of a wide range of spectrum extending from home apparatuses and cell phones to medical apparatus and transport controllers. They are commonly portrayed by their real-time behavior and must satisfy strict requirements on reliability and accuracy.
"An embedded system is a system that has embedded software and computer hardware which makes it a system dedicated for an application or specific part of an application or product or part of large system" [1].
embedded system consist of three main components:

1. It has hardware.

2. It has main application software which performs set of numerous tasks.

3. It has a real time operating system (RTOS) that manages the application software and gives a mechanism to let the processor run a process according to scheduling and do the context-switch between the multiple processes (tasks).

Software systems are the complex system and the functional requirements of software application are very much important. But, they are not the only concern. Performance evaluation of the system is also a crucial task. Petri net is an adequate formal method for modeling and analysis of various embedded system.
In the analysis of real time system, task completion time is the crucial part which

must be satisfied. The worst case and best case completion time of the system must be guaranteed before putting the system into utilization. The lack of complete knowledge of the system results in uncertainties which may lead to extend the completion time of the system. In the analysis of concurrent system, Petri net has been used widely and effectively. In the process of software development, after completing requirements analysis and achieving a requirement specification document, appropriate style of software architecture of the system is supposed to be identified from the very inception phase. Identification of prominent architectural style activity plays a significant role for proper implementation of the system. Non-functional requirement such as performance, reliability, security and so on are influenced by proper identification of an architecture style.

Petri nets represent the system behavior graphically as well as mathematically. The modeling and analysis of the system is done by the theoretic aspect of the Petri nets while picturization of designed system state changes is done by graphical aspect. Consequently, various dynamic event-driven systems are modeled by Petri nets.

Petri Net as a mathematical formalism acknowledges the performance of the behavior of real time systems and helps to design software architecture. There are various Petri net tools like CPN, PIPE, SPNP, SNOPPY, TIMENET, TAPAAL etc. for portraying and studying the behavior of the system.

1.2 Petri Nets

A Petri Nets is mathematical model used to describe any system. It is a directed bipartite graph defined as a 6 tuple, (P,T,A,I(.),O(.),m_0), where,

- P represents a set of places.
- T represents a set of transitions.
- A represents a set of arcs.
- I(.) represents the input functions, maps transitions to places.

- O(.) represents the output function, that maps places to transitions.
- m_0 represents the initial marking of the net, where marking is the number of tokens in each places.

Petri nets commonly represented in a graphical manner. Places are represents as circle, transitions are illustrated as bars and arcs are represented as arrow. Tokens are depicted as dots inside a place. Which describes the number of resources acquired at a particular state. The transitions are said to be enabled when the input places are having tokens. Petri Nets are used to acquire the behavior of lot of real world circumstances. The main aspect of Petri Nets model is the representation of concurrent execution of activities.

1.3 Motivation

Correctness performs a key part in numerous embedded systems. The failure of such system results in a loss of both human lives and money as we get highly dependent on them. So, reliability and safety are the most crucial benchmark for the embedded system. The implementation of the system fulfill its requirement is proved formally by analytical and mathematical techniques provided by formal methods

1.4 Objective

Modeling of the system is the crucial part of a design flow stage. One of the objectives of this thesis is to define a formal representation of the system proficient of acquiring essential properties of the embedded systems. It must be, accurate enough so that the interpret and deal it easily. Since correctness and reliability are getting to be progressively essential for embedded systems, it is also aimed at verifying such system by using formal methods.

1.5 Organization of Thesis

The thesis is structured as follows: Chapter 2 expresses the literature survey done for this thesis. It includes the work done in the area of Petri nets. Chapter 3 illustrates the design representation of the embedded system using Petri net tools.For this purpose, a case study of an Elevator Control System is taken and modeled. Chapter 4 discussed the performance evaluation of the modeled system using Stochastic Petri nets. Chapter 5 discusses the model verification using Computation tree logic (CTL) to check the correctness of the model. Finally chapter 6 concludes the work done with the scope of future work.

Chapter 2

Literature Review

Chapter 2

Literature Review

The essential part in design methodology is the modeling of the system. Numerous computation models have been developed in the literature to represent to advanced systems. These models incorporate an expansive extent of styles, aspects, and provision areas. Especially in embedded system design, a mixed bag of models have been produced and utilized as representation of the system.

This chapter exhibits related work done in the field of modeling, performance evaluation and verification of embedded system.

Petri net was developed by Carl Adam Petri in 1962 [2]. His work has been further elaborated by T. Murata, in which he analyzed various structure of Petri nets, their markings and executions [3].

Jensen proposed the basic concept of Colored Petri nets [4]. The tool used for Colored Petri Net is CPN tool. It provides a graphical modeling of the system. It verifies all the constraints or conditions which are necessary to be checked for deployment of better software. CPN is a tool which provides edition, simulation and analyzsis of timed and untimed hierarchical Colored Petri nets.

Huafeng Zhang proposed "Modeling a Heterogeneous Embedded System in Colored Petri Nets" in 2014 [5]. In his paper, a heterogeneous system of Vehicle protocol which is composed of two subsystems is introduced and pointed out a potential defect in that system caused by an interface mismatch. Then, a state based approach is applied to verify analysis of the system.

Nabil R. Adam proposed " Modeling and Analysis of Workflows Using Petri Nets" in which he has demonstrated the use of PN as an effective tool for modeling

workflows at a conceptual level and then analyzing them [6]. He has developed a PN-based approach that uses several structural properties such as the P-invariant, siphon, min/max analysis, etc. for identification of inconsistent dependency specification in a work flow and testing for its safe execution.

Wil M.P. van der Aalst proposed Strategies for "Modeling Complex Processes using Colored Petri Nets" in 2013 in which he used a running example to explain several design patterns for modeling in terms of CPNs [7]. These patterns guide users in modeling complex processes that require interplay of control-flow and data-flow.

Olivier and Roy introduced an approach to implement a distributed monitor of real-time system properties, and then introduced a new formalism, adaptive Petri nets, that allows to model such complex, distributed and real time systems [8].

The performance analysis of the model illustrates the behavior of the system. Falko Bause proposed the concept of stochastic Petri nets with various examples [9].In Stochastic Petri nets (SPN), random firing delays are attached to the transition. The SPN is used for performance analysis of the system by Markovian techniques.

Michael K. Molloy proposed the performance analysis of the system using Stochastic Petri nets [10]. They used Stochastic Petri net for performance analysis of alternating bit protocol.

Bernardi proposed a structural performance evaluation methodology for Timed Petri nets (TPNs) and their stochastic extensions [11].

Marcelo H. Cintra presented "A Simulation Technique for Performance Analysis of Generic Petri Net Models of Computer Systems" in which he presented a simulation algorithm tp observe that the simulator performed reasonably well, even on a modest machine [12].

Correctness plays an important role in many embedded systems. Model checking is an approach used to decide whether the model of a system fulfill a list of essential properties. In the area of formal verification, many mechanism have also been presented.

Xudong He proposed a methodology of associating time predicate transition nets

and flourished a real-time first order temporal logic for specification and verification of real-time systems [13].

W.M.P. van der Aalst proposed a methodology to verify the business process using Petri nets [14]. In which he verified the liveness and boundedness of work flow net using petri nets.

Hanifa Boucheneb and Rachid Hadjidj proposed model verification techniques of time Petri nets [15]. They used temporal logic model checking to represent the behavior of a system.

Chapter 3

The Design Representation

Chapter 3

The Design Representation

3.1 Colored Petri nets

3.1.1 Introduction

Since Petri nets support only one type of tokens, so, the graphical representation of the system will become very complex for analysis. Hence Colored Petri nets (CPNs) has been introduced in which a type called as *color* is connected to a token. Colored Petri nets are graphical languages for developing models of real time systems and analyzing their properties [4]. CPN is formally defined as 9 tuple, CPN = ($\sum$, P, T, A, N, C, G, E, I) where,

- $\sum$ is a finite set of non-empty types, called color sets.
- P represents a finite set of places
- T represents a finite set of transitions
- A represents a finite set of arcs
- N represents a node function
- C represents a color function.
- G represents a guard function.
- E represents an arc expression function.
- I represents an initialization function.

The tool used for Colored Petri Net is CPN. It provides a graphical modeling of the system. It verifies all the constraints or conditions which are necessary in the analysis phase. It helps in analyzing timed and un-timed hierarchical Colored Petri nets.

3.2 Case Study: Elevator Control System

A case study of Elevator Control System is taken and modeled with Colored Petri nets [16]. Elevator control system is a software that manages the multiple elevators in a building in order to facilitate in transportation from one floor to another floor. An elevator control system has many typical aspects of time and critical safety requirements. The elevator control system contains certain components such as request button, sensor and door which function according to the passenger's requirements. The problem in detail:

- There are F floors in a building.
- There is one elevator that we are concerned with.
- There are elevator service requests coming from people on each floor.
- The request comes at a particular time. The request is to go from one floor (source) to another(destination).

Figure 3.1 shows the modeling of Elevator Control System using Colored petri nets. To validate the constraints applied on Elevator Control System, functions and guard condition have been used. initially, there are requested floor button and current floor button. First, the passenger will request for the desired floor. The request will be transmitted to the controller and it will be matches with the current floor. if the requested floor and the current floor are different then only the request will be processed. The time parameters are also associated with the request number to show the timing of a particular request. The requests are served in a First Come First Serve (FIFO) manner. The arguments which are passed to the functions are as follows:

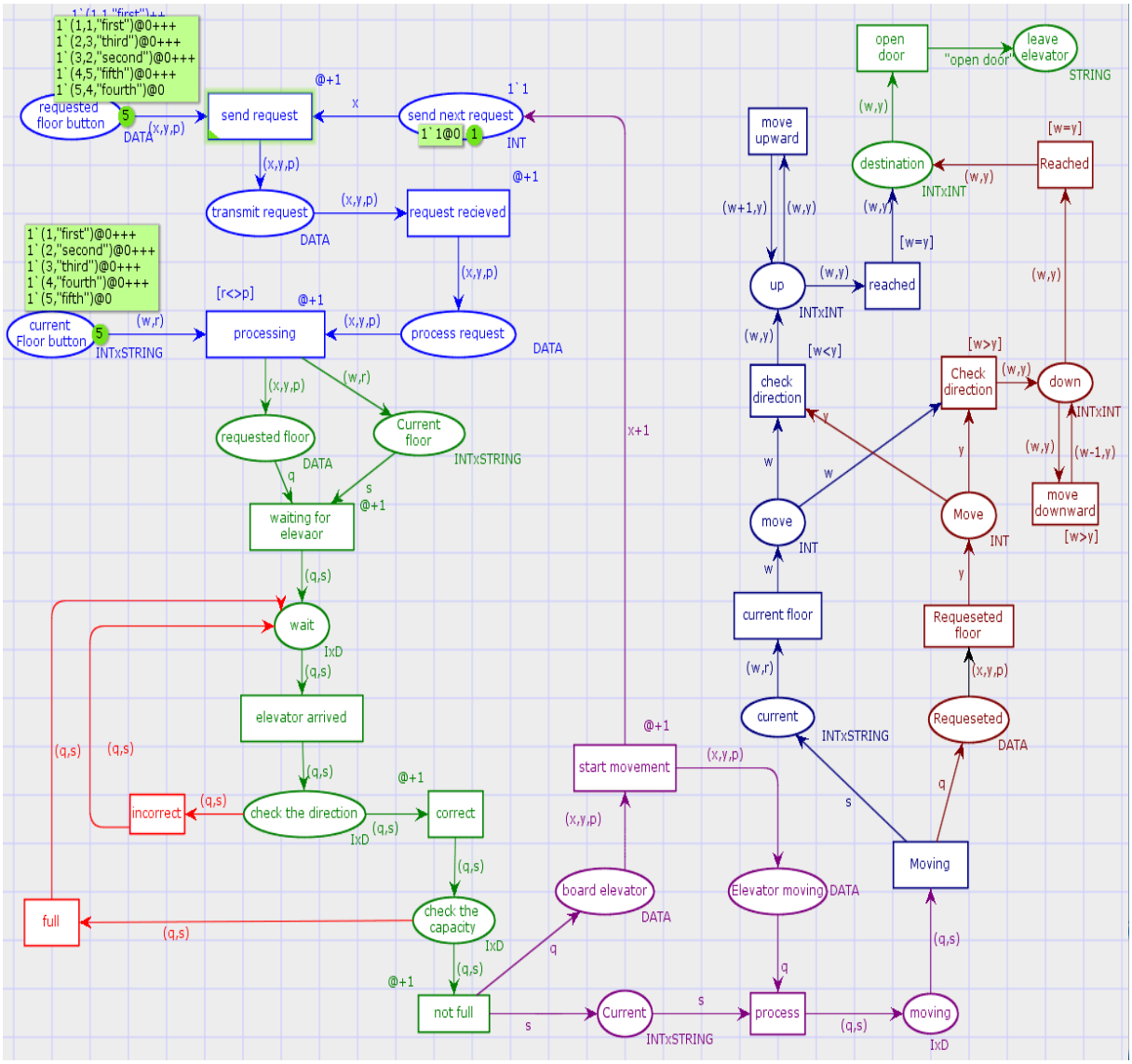

Figure 3.1: Modeling of Elevator Control System using CPN

q- Requested floor button

s- Current floor button

x- Request number

y- Requested floor number

p- Requested floor

w- Current floor number

r- Current floor

Algorithm 1: send request function

1. **Request number:** It represents the no. of requests.
2. **requested floor:** It represents the current request for floor.
3. **current floor:** It represents the current floor number.
   Input: Requested number,requested floor, current floor.
4. **if** current floor ! = requested floor **then**
5. Call **process request**
6. **else**
7. Call **send next request**
8. **end if**

Algorithm 2: Process request function

1. **Request number:** It represents the no. of requests.
2. **requested floor:** It represents the current request for floor.
3. **current floor:** It represents the current floor number.
   Input: Requested number,requested floor, current floor.
4. wait for elevator to arrive
5. elevator arrived
6. check the direction
7. **if** correct **then**
8. Call **check the capacity**

9. **else**
10. Call wait
11. **end if**
12. **if** capacity = not full **then**
13. board elevator and Call **start movement**
14. **else**
15. Call wait
16. **end if**

Algorithm 3: start movement function

1. **Request number:** It represents the no. of requests.
2. **requested floor:** It represents the current request for floor.
3. **current floor:** It represents the current floor number.
   Input: Requested number,requested floor, current floor.
4. Call check direction
5. **if requested floor number > current floor number then**
6. Call **move upwards**
7. **else**
8. Call **move downwards**
9. **end if**

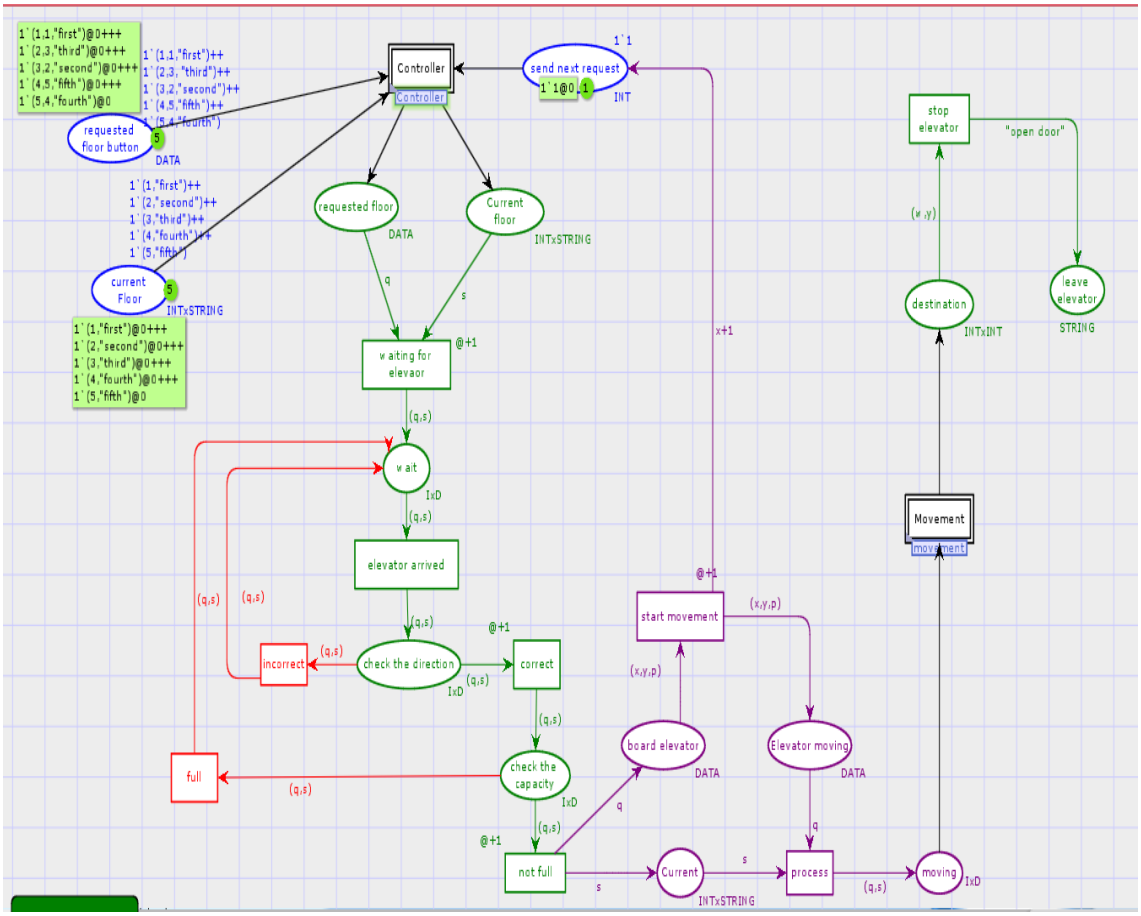

Figure 3.2: Hierarchical Representation of Elevator Control System

Figure 3.2 shows the Hierarchical representation of the Elevator Control system. The sub models of the main page are "Controller" and "movement".

Figure 3.1 shows the modeling of elevator Control System using Colored petri nets. To validate the constraints applied on Elevator Control System, functions and guard condition have been used. initially, there are requested floor button and current floor button. First, the passenger will request for the desired floor. The request will be transmitted to the controller and it will be matches with the current floor. if the requested floor and the current floor are different then only the request will be processed. The time parameters are also associated with the request number to show the timing of a particular request. The requests are served in a First Come First Serve (FIFO) manner. If the elevator is in moving state,

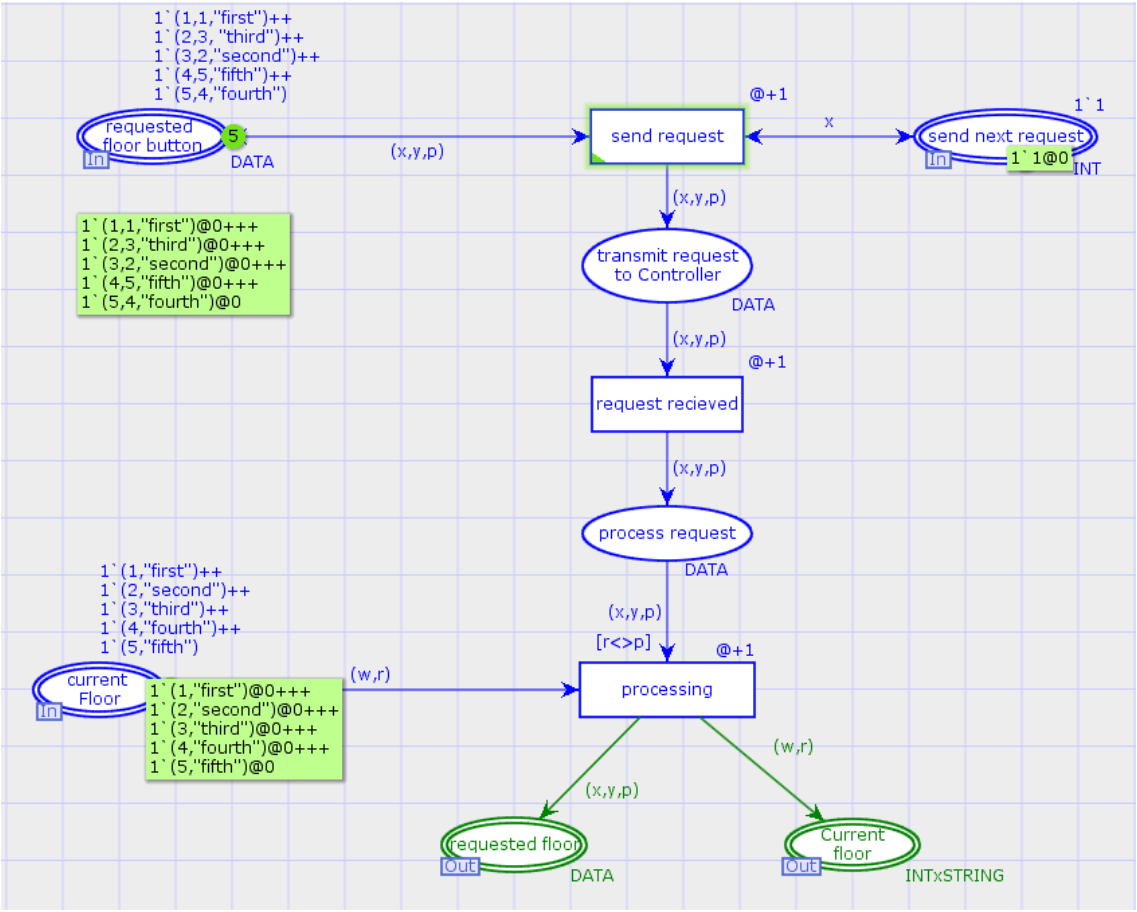

Figure 3.3: Processing of Request

the next request can not be served it must be queued. When the elevator finishes its current request then only the next request can be served.

When the controller receives the request, it goes into the “waiting” state. when the elevator arrives, the controller checks the direction and capacity of the elevator. If direction is correct and the elevator is not full, then the token moves to the “board elevator” state, otherwise again the token moves to the “waiting” state. After “board elevator” state, the token moves to the “moving state” and “send next request”. The controller checks the direction of the request whether it is upward or downward. If the requested floor number is less than current floor number, then the token passes to the ”move downward” state and if the requested floor number is greater than current floor number, then the token passes to the ”move upward” state. while moving the elevator, there is a guard condition that checks whether current floor number is equal to requested floor number. If the guard condition

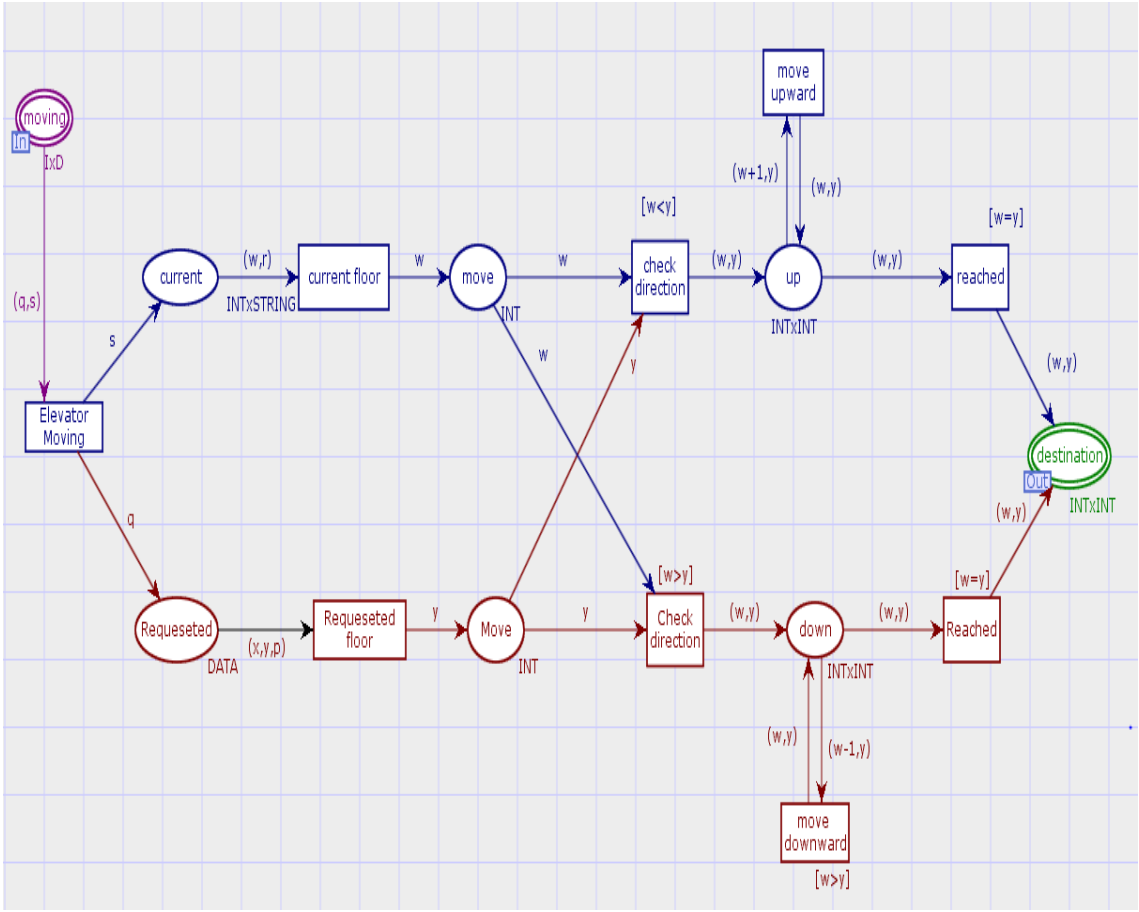

Figure 3.4: Moving elevator Upward and Downward Direction

is satisfied, the token will move to the "destination" state. After reaching the destination, the door opens and the token moves to the "leave elevator" state.

Algorithm 4: moving upward and downward function

1. **Request number:** It represents the no. of requests.
2. **requested floor:** It represents the current request for floor.
3. **current floor:** It represents the current floor number.
   Input: Requested number,requested floor, current floor.
4. Call check direction
5. **if direction = upward then**

6. **do** current floor number + 1
7. **while** current floor number ! = requested floor number
8. **end do while**
9. **else**
10. **if direction = downward then**
11. **do** current floor number − 1
12. **while** current floor number ! = requested floor number
13. **end do while**
14. **end if**
15. **end if**

When the destination is reached, the door will be opened and passengers will be delivered to the desired floor. After delivering the passengers, the door is closed and the elevator will go in an idle state.

3.3 Petri net Markup Language (PNML)

3.3.1 Introduction

The PNML is an XML-based interchangeable format for various kinds of Petri nets, by which different tools can exchange Petri net models among each other [17]. Due to generic feature of PNML, a procedure for defining own type of Petri net is supported by PNML. This feature of PNML is known as Petri net type definition(PNTD).

PNML manages the principle of flexibility, compatibility and unambiguity. Flexibility means any type of Petri net with is its particular properties can be represented by PNML. PNML supports the labeling of graph where all the necessary information are saved in the label and the label is then connected to the net by an arc.

PNTD determines a valid label for a specific Petri net type which assigns a fixed type to make the description unambiguous.

Compatibility feature of PNML states that a large number of information can be interchange among several types of Petri nets. To attain compatibility, labels must be defined with a particular meaning. All Petri net files support the specific type i.e., Uniform Resource Identifier (URI) by which the syntax is uniquely identified. Each PNTD has a prescribed semantics that is identified by other tools which use it.

There are various tools of PNML which support xml based interchangeable format, such as ePNK, Rnew, Design/CPN, PEP etc.

3.3.2 ePNK tool of PNML

ePNK is the eclipse based Petri net Kernel tool of PNML. ePNK constructs a data framework for Petri nets which is synonymous to that of PNML. Based on the Petri net type, several labels can be attached to place, arc, transition, or even the net. An extension point is provided by the ePNK so that any new petri net can be attached to it without interfering the code of ePNK. To define a new Petri net type, the designer has to give a class diagram which describes the concept of

the new Petri net type with a correspondence of xml syntax. This type can then be attached into the ePNK tool.

3.4 Case Study: Automatic Teller Machine (ATM)

A case study of "Automatic Teller Machine " is taken and modeled with ePNK tool of PNML [18]. ATM is geographically distributed across all branches of the owner bank and all are connected to a central server. ATM allows the customers to access their account balance, transfer money between accounts, and withdraw money after validating their identity by checking their entered personal identification number(PIN) with the central bank server.

Figure 3.5 shows the modeling of ATM system using Petri net markup language. The model validates all the constraints of specified requirements. when a user wants to use his card on ATM, he must have the valid card which the bank can recognize, then he has to enter the valid PIN. The used is allowed to enter the PIN only three times in case of entering wrong PIN. If the PIN is right, he can proceed the operations, otherwise his card will be rejected. After entering the right PIN, the user is able to do the operations whichever he wants such as cash withdrawal, balance inquiry, mini statements, money transfer an so on. After the successful operation, the card is dispensed from the ATM.

After validating the model, an xml code is generated. The xml code can be imported by other petri net tools which support PNML format.

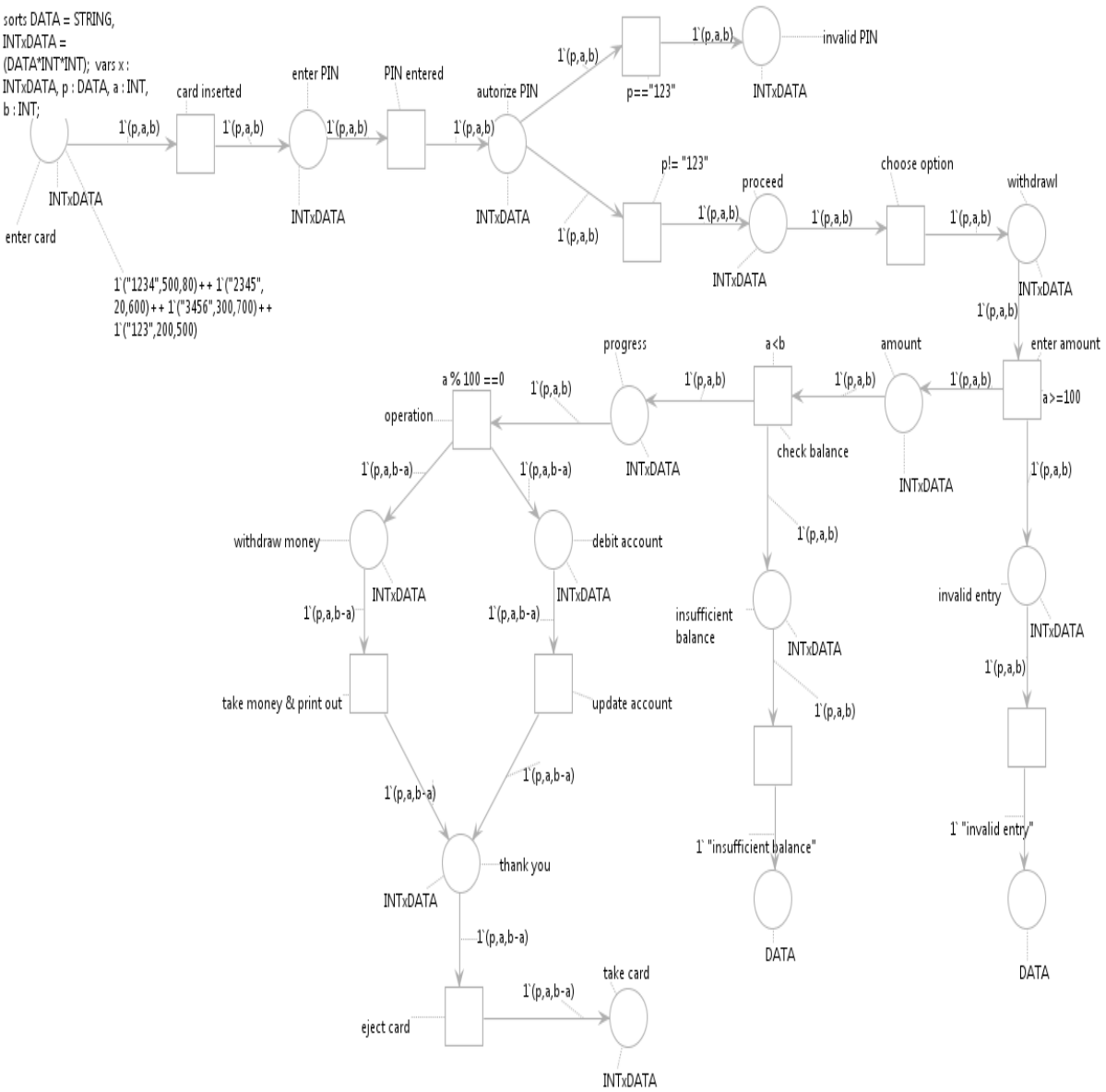

Figure 3.5: Modeling of ATM system using ePNK tool of PNML

3.5 Summary

The modeling of Elevator Control System is illustrated using Colored Petri nets. The model captures the essential features of the Elevator Control System. The timing constraints are associated with the requests which are coming in random order and the requests are served in First Come First Serve (FIFO) manner. The algorithms are proposed to satisfy the necessary constraints of the Elevator Control System.

Another modeling representation of embedded system i.e., ATM system is illustrated using Petri net markup language (PNML). PNML is an xml based interchangeable language by which the model can be imported to other PN tools.

Chapter 4

Performance Evaluation of the System

Stochastic Petri nets

Platform Independent Petri net Editor

Implementation and Result

Summary

Chapter 4

Performance Evaluation of the System using Stochastic Petri Nets

4.1 Introduction

Problems in a system which includes both hardware and software sub-system which perform a specific task or mission during a specific time in a specific environment are dealt by system reliability. Stochastic Petri nets (SPN) have great power to analyze the performance of the model using Markov chain. The transitions whose firing is an atomic operation, a random firing delay is associated with those transitions. A stochastic methodology is a numerical model helpful for the portrayal of phenomena of a probabilistic nature as a function of parameter which is associated with time. The execution policy specify the firing of transition when more than one transitions enabled at time in SPN model. The execution policies are race policy and the pre-selection policy. In race policy the transitions whose firing time elapse first, will fired first. In pre-selection policy, the probabilistic distribution function determines that which transition will fire first from the enabled transitions . A SPN model operates on race policy.

4.1.1 Generalised Stochastic Petri Nets

Ajmone Marsan et al proposed the concept of Generalized Stochastic Petri Nets (GSPN) [19]. GSPN is an advancement of SPN. Two classification of transitions are supported by GSPN i.e., immediate and timed transition. In immediate transition, the firing time of the transition is zero. In timed transition, the firing rate of the transition is exponentially distributed. If both the timed transition and immediate transition activated simultaneously, the transition which fire first is the immediate transition. If the immediate transition becomes enable in any marking then that marking is known as "vanishing marking" and if timed transition becomes enable in any marking then the marking is known as "tangible marking". The SPN model becomes markovian in nature when it has both the immediate and timed transition. In markov process, the future state depends only on the present state. After that the SPN model can generate the corresponding markov chain [20]. The markov chain can then be used to measure various performance parameters.

A standard example of a Stochastic Petri net (SPN) is depicted in Figure 4.1. In Figure 4.1, at marking $M_0 = (1, 0, 0, 0, 0)$, transition t_1 is activated . The firing rate of t_1 is λ_1 i.e., exponentially distributed. The average time for firing t_1 is $1/\lambda_1$. When t_1 fires, the marking will result in $M_1 = (0, 1, 1, 0, 0)$. t_2 and t_3 are both enabled at marking M_1. If t_2 fires first, the marking will change to $M_2 =$ (0,0,1,1,0) and if t_3 fires first, the marking will become $M_3 =$ (0,1,0,0,1). So, the next marking will depends on which transition "win the race".

The probability of firing t_2 first is demonstrated as:

$$\begin{aligned} P\left[t_2 fires first at M_1\right] &= P\left[X_2 < X_3\right] \\ &= \int_0^\infty \left(\int_0^x \lambda_2 e^{-\lambda_2 y} dy\right) \lambda_3 e^{-\lambda_3 x} dx \\ &= \int_0^\infty \left(1 - e^{-\lambda_2 x}\right) \lambda e^{-\lambda_3 x} dx \\ &= \lambda_2/(\lambda_2 + \lambda_3) \end{aligned} \tag{4.1}$$

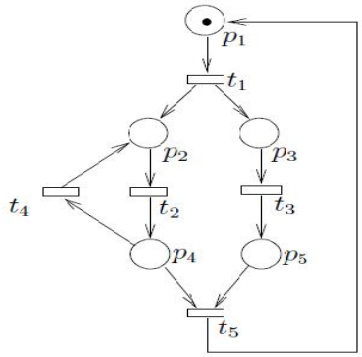

Figure 4.1: Stochastic Petri net

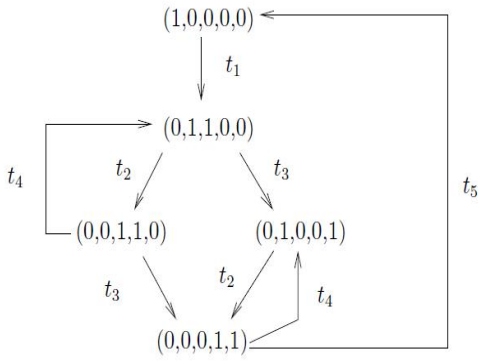

Figure 4.2: Reachability graph of SPN

and similarly, the probability of firings of t_3 can be represented as:

$$P\left[t_3 fires first at M_1\right] = \lambda_3/(\lambda_2 + \lambda_3) \tag{4.2}$$

The sojourn time in M_1 is given by the minimum of the independent and exponentially distributed firing times of both transitions,

$$\begin{aligned} P\left[min\left(X_2, X_3\right) \leq x\right] &= P\left[X_2 \leq x \, or \, X_3 \leq x\right] \\ &= 1 - P\left[X_2 > x \, and \, X_3 > x\right] \\ &= 1 - e^{-\lambda_2 x} e^{-\lambda_3 x} \\ &= 1 - e^{-(\lambda_2 + \lambda_3)} \end{aligned} \tag{4.3}$$

Thus the sojourn time in M_1 is also exponentially distributed with parameter $(\lambda_2 + \lambda_3)$.

Figure 4.2 illustrates the reachability graph of Figure 4.1. Markov Chain can be obtained by considering each marking in the reachability graph and attaching the firing rate λ_i of transition i as an arc label to each transition.

The Markov chain of an SPN can be obtained from the reachability graph as follows:

R(PN) represents the reachability set of MC state space and the transition rate from state M_i to state M_j is given by $q_{ij} = \lambda_k$. If several transitions are connected from M_i to M_j then q_{ij} represents the sum of the rates of those transitions.

Similarly $q_{ij} = 0$ if there is no transitions connected from M_i to M_j.

The steady state distribution π of the MC is attained by solving the following linear Equations

$$\pi Q = 0 \tag{4.4}$$

$$\sum_{i=1}^{s} \Pi_j = 1 \tag{4.5}$$

where,

Q is the square matrix $Q = (q_{ij})$ *of order* $s = |R(PN)|$. From the vector $\pi = (\pi_1, \pi_2, ..., \pi_s)$ the following performance measures may be computed.

Probability of being in a subset of markings:

Let $B \subseteq R(PN)$ comprises the markings in a specific SPN. Then the probability of belonging in a state of the related subset of the Markov chain is given by:

$$P[B] = \sum_{M_i \epsilon B} \pi_i \tag{4.6}$$

Average number of tokens:

Let $B(p_i, n)$ is the subset of $R(PN)$ for which the number of tokens in a place p_i is n, i.e., $B(p_i, n) = M \epsilon R(PN) | M(p_i) = n$. Then the mean number of tokens in place p_i is given by:

$$\bar{m_i} = \sum_{n=1}^{\infty} (nP[B(P_i, n)]) \tag{4.7}$$

Probability of firing transition t_j :

Let EN_j be the subset of R(PN) in which a given transition t_j is enabled. Then, the probability r_j for transition t_j firing next is given by:

$$r_j = \sum_{M_i \epsilon EN_j} \pi_i (\lambda_j \setminus (-q_{ij})) \tag{4.8}$$

where $(-q_{ii})$ is the sum of transition rates out of M_i.

Throughput at a transition t_j :

The throughput at a timed transition is given by it's average number of firings at steady state:

$$\bar{d_j} = \sum_{M_i \epsilon EN_j} \pi_i \lambda_j \tag{4.9}$$

The performance of the model can be investigated on the basis of the values of these parameters.

4.2 Platform Independent Petri net Editor (PIPE)

PIPE is platform independent tool for modeling and analyzing Petri nets using Stochastic Petri nets [21]. It is developed thoroughly in Java which provides security of platform independence feature and presents a standard, adaptable graphical user interface for analyzing Petri net model. PIPE also provides a convenient work space for analyzing module to check behavioral assets and produce performance demography.

4.3 Implementation and Result

In this work, a case study of Elevator Control System is taken and modeled with PIPE tool. After modeling the system, performance analysis is carried out to check the stochasticity of the model.

Figure 4.3 shows the modeling of Elevator Control System using PIPE tool. The initial state in the model is "idle" state. when user arrives at a floor and presses the button, the token in the model moves to "wait" state. When the elevator arrives, user checks whether the elevator is moving in correct direction or not. If the elevator is moving in right direction then the token moves to the next state otherwise the token goes back to the "press button" state. If the elevator is moving in correct direction then it checks the capacity of the elevator. If the elevator is full then the token goes back to the "wait" state otherwise it goes to

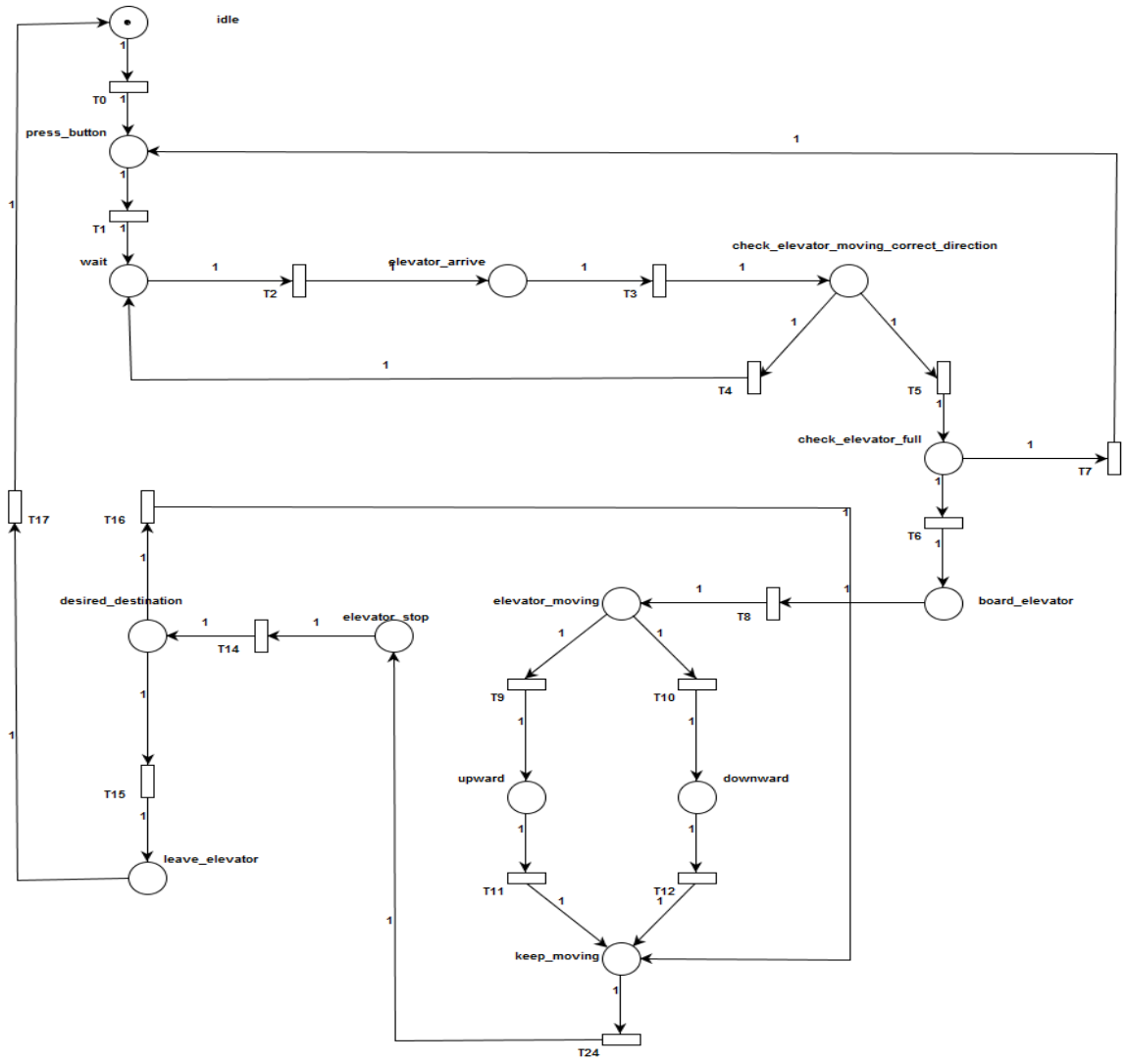

Figure 4.3: Modeling of Elevator Control System using PIPE

the next state "board elevator". After this the token gets passed to the "elevator moving" state. At this state, The controller checks the direction and move the elevator upward or downward according to the request. If the elevator reaches the desired floor then token gets passes to the "leave elevator" state, otherwise token again moves to the "moving elevator" state. When user leaves the elevator the token again moves to the idle state.

After the modeling of Elevator Control System, the performance evaluation has been carried out using PIPE tool. The PIPE tool analyzes steady state distribution of tangible states, mean number of tokens in each place, token probability

density, throughput of timed transition and sojourn time of tangible states.

4.3.1 Result

The performance analysis of the Elevator Control System ara as follows:

Table 4.1: Steady State Distribution of Tangible States

Marking	value
M_0	0.04445
M_1	0.08889
M_2	0.17778
M_3	0.17778
M_4	0.08889
M_5	0.04444
M_6	0.04444
M_7	0.02222
M_8	0.02222
M_9	0.02222
M_{10}	0.08889
M_{11}	0.08889
M_{12}	0.04444
M_{12}	0.04444

Table 4.2: Average Number of Token on a Place

Place	Number of Tokens
idle	0.04444
press_button	0.08889
wait	0.17778
elevator_arrive	0.17778
check_elevator_moving _correct_direction	0.08889
check_elevator_full	0.04444
board_elevator	0.04444
elevator_moving	0.02222
upward	0.02222
downward	0.02222
elevator_stop	0.08889
desired_direction	0.04444
leave_elevator	0.04444
keep_moving	0.04444

Table 4.1 shows the steady state distribution of tangible states. Tangible states are those states which are associated with the timed transition. Table 4.2 shows the average number of tokens present in a place. It can be calculated by the Equation 4.7 such as:

$$\bar{m_i} = \sum_{n=1}^{\infty} (nP\left[B\left(P_i, n\right)\right])$$

Table 4.3 specifies the token probability density.

Table 4.4 shows the throughput of timed transition. The throughput at a timed transition is expressed by it's average number of firings at steady state. It can be measured by the Equation 4.9 i.e.,

$$\bar{d_j} = \sum_{M_i \epsilon EN_j} \pi_i \lambda_j$$

Figure 4.4 shows the reachability graph of the model. The firing of enabled transition is responsible for changing the state of the model. The sequence of

Table 4.3: Token Probability Density

Place	$\mu = 0$	$\mu = 1$
idle	0.95556	0.04444
press_button	0.91111	0.08889
wait	0.82222	0.17778
elevator_arrive	0.82222	0.17778
check_elevator_moving _correct_direction	0.91111	0.08889
check_elevator_full	0.95556	0.04444
board_elevator	0.95556	0.04444
elevator_moving	0.97778	0.02222
upward	0.97778	0.02222
downward	0.97778	0.02222
elevator_stop	0.91111	0.08889
desired_direction	0.0.95556	0.04444
leave_elevator	0.95556	0.04444
keep_moving	0.91111	0.04445

Table 4.4: Throughput of Timed Transition

Transition	Throughput
T_0	0.04445
T_1	0.08889
T_2	0.17778
T_3	0.17778
T_4	0.08889
T_5	0.08889
T_6	0.04445
T_7	0.04445
T_8	0.04445
T_9	0.02222
T_{10}	0.02222
T_{11}	0.02222
T_{12}	0.0.02222
T_{14}	0.08889
T_{15}	0.04445
T_{16}	0.04445
T_{17}	0.04445
T_{24}	0.08889

Table 4.5: Sojourn Time for Tangible States

Marking	value
M_0	1
M_1	1
M_2	1
M_3	1
M_4	0.5
M_5	0.5
M_6	1
M_7	0.5
M_8	1
M_9	1
M_{10}	1
M_{11}	1
M_{12}	0.5
M_{13}	1

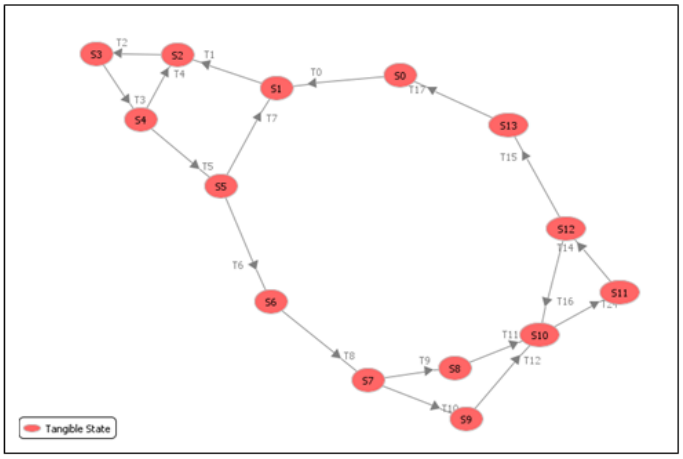

Figure 4.4: Reachability graph of SPN's Underlying

firing results in getting a sequence of marking. This sequence of marking can be portrayed as reachability graph.

Table 4.6: Petri net State Space Analysis Result

Bounded	True
Safe	True
Deadlock	False

Table 4.6 illustrates the state space analysis of the model. By the analyzing the model, it is observed that the model is bounded, safe and deadlock free. A model is called as bounded if the number of token in each place does not cross a finite number k, where k is the natural number.

The performance of the model can be investigated on the basis of the values of these parameters.

4.4 Summary

The modeling of Elevator Control System is illustrated using PIPE tool. The performance analysis is carried out to evaluate the model based on some performance parameter such as token probability density, average number of token in each place, throughput of timed transition and so on. By analyzing the model it is observed that the model is bounded, safe and deadlock free.

Chapter 5

Model Verification using TAPAAL Tool

Chapter 5

Model Verification using TAPAAL tool of Petri nets

5.1 Introduction

Model checking and verification is the essential part of developing any system to check whether the desired behavioral properties of the system are met or not. Ensuring the correctness of the system and design validation of the system is the very challenging task. For the verification purpose of models, temporal logics are used to check the correctness of the models.

5.1.1 Temporal Logic

Temporal logics are used to check how the truth of the arguments change over the period of time. Temporal logics are mounted with the temporal operators. They are used to determine the supposed behavior of the systems.Depending upon the model of time, there exist divergent types of temporal logics. In this part, the work is focused on computation tree logic (CTL). CTL is a representative of temporal logic and used to verify different models.

Computational Tree Logic

Computational Tree logic (CTL) is based on propositional logic of forking time. Forking time states that time may divided into more than one future state using separate model of time. CTL supports temporal operators such as G, F, X, U, A, E where,

G represents globally

F represents in future

X represents next time

U represents until

A represents all computation path

E represents some computation path.

The temporal operator G, F, X, and U preceded by A or E.

An unfolded state graph is pictured by CTL. The nodes of the state graph represents the possible state where the system may reach. In the state graph, there are some shaded nodes also exist which represents that certain property p holds in the system or not. The root node in the graph represents the initial state of the system. For example, AF p occupied in the initial state if for every possible path property p is satisfied by at least one state starting from the initial state. The other temporal operator can be explained in the same way.

5.1.2 TAPAAL Tool

TAPAAL tool is used for model verification to check the properties of the system to be satisfied by the model [22]. TAPAAL tool provides an editor, a simulator and a verifier for Timed-Arc Petri Nets (TAPN) [23]. In TAPAAL tool, an *age* (a real number) is joint to each token. there is a procedure to add the time interval to limit the ages of the tokens which is responsible for the firing of the transitions. The eminence of using this model is decide whether the particular model is bounded and coverable or not. TAPAAL also provide a mechanism to verify the model to check whether certain properties are satisfied by the model or not. TAPAAL use UPAAL engine at the back end to verify the model using queries in the constructed net [24]. A graphical query dialogue is provided by TAPAAL tool to write the queries. The queries are written with the help of computation tree logic which consist of EF, AG, EG, and AF temporal operators where E represents "there exist an execution", A represents "for all execution", F represents "finally" and G represents "globally". The combination of these operators can be elaborated as:

EF = There exist some reachable marking that satisfies the given condition.

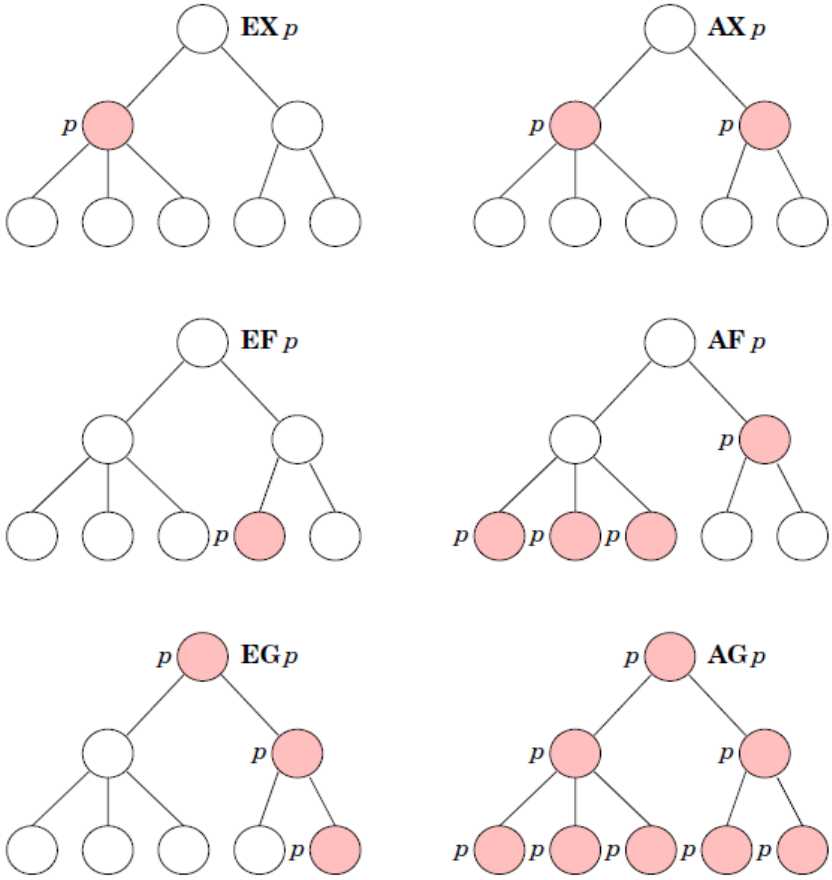

Figure 5.1: CTL Temporal Operators

EG = There exist a trace on which every marking satisfies the given condition.

AF = On all trace there is eventually a marking that satisfies the given condition.

AG = All reachable marking satisfy the given condition.

5.2 Implementation and Result

Figure 5.2 shows the model of Elevator control system using TAPAAL tool. The TAPAAL tool provides a graphical editor for drawing Timed-Arc Petri Net (TAPN) models. It also offers simulator for examining the modeled nets and a verification domain that accordingly answers logical queries [23]. With the help of TAPAAL tool, boundedness of the model can also be checked. The boundedness of the system can be described as that the number of tokens present in each place must not be greater than a finite number k.

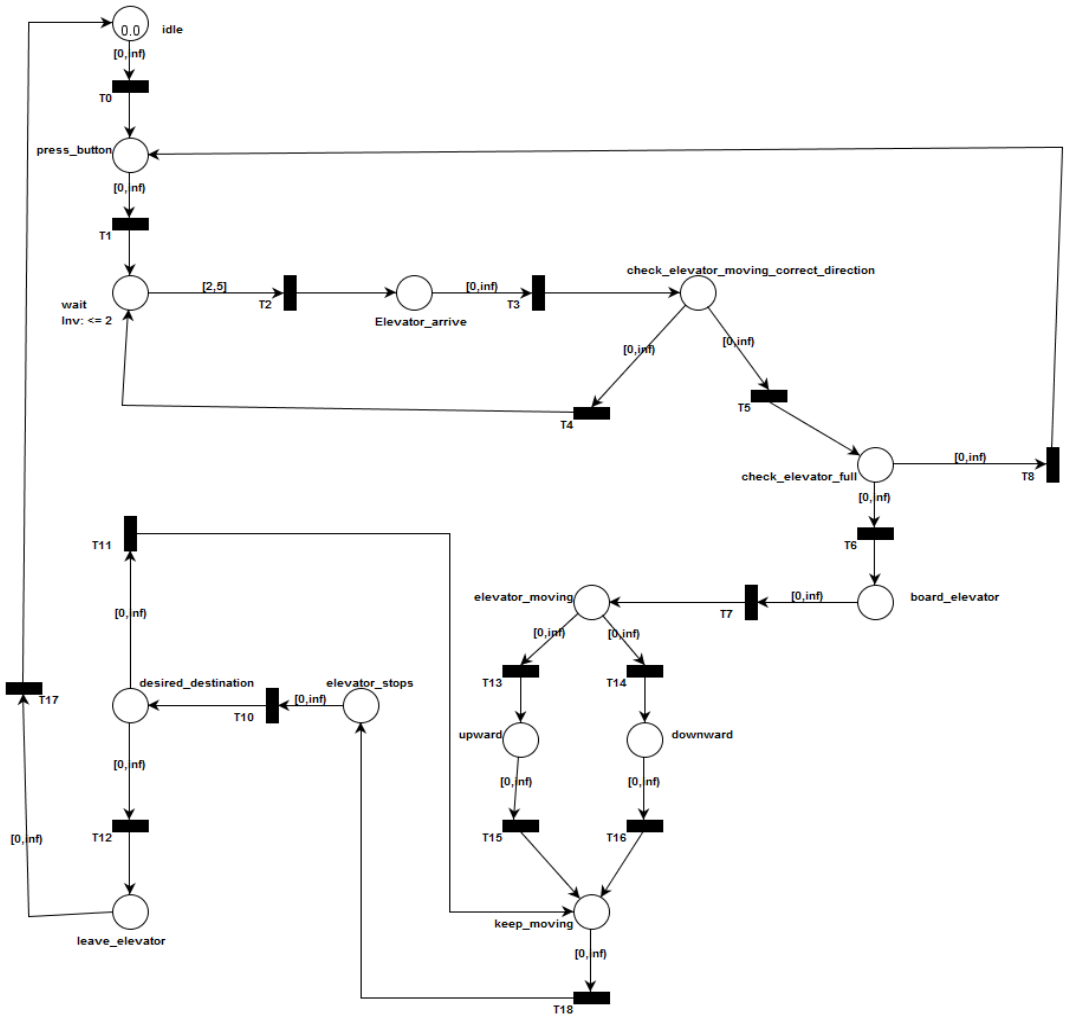

Figure 5.2: Modeling of Elevator Control System using TAPAAL

Figure 5.3 shows the query editor to evaluate the query and to check whether the model is correct or not. The query written in the query editor is :
EF(TAPN.board_elevator=1 and TAPN.press_button=0)
The query checks that whether there exist some path for which board_elevator sate is enabled and press_button state is disabled.
TAPAAL's verification module allows the user to verify safety and liveness queries in the designed net.

Figure 5.4 illustrates the result of the query. If the query is correct and the properties are satisfied, it means that the model is correct.

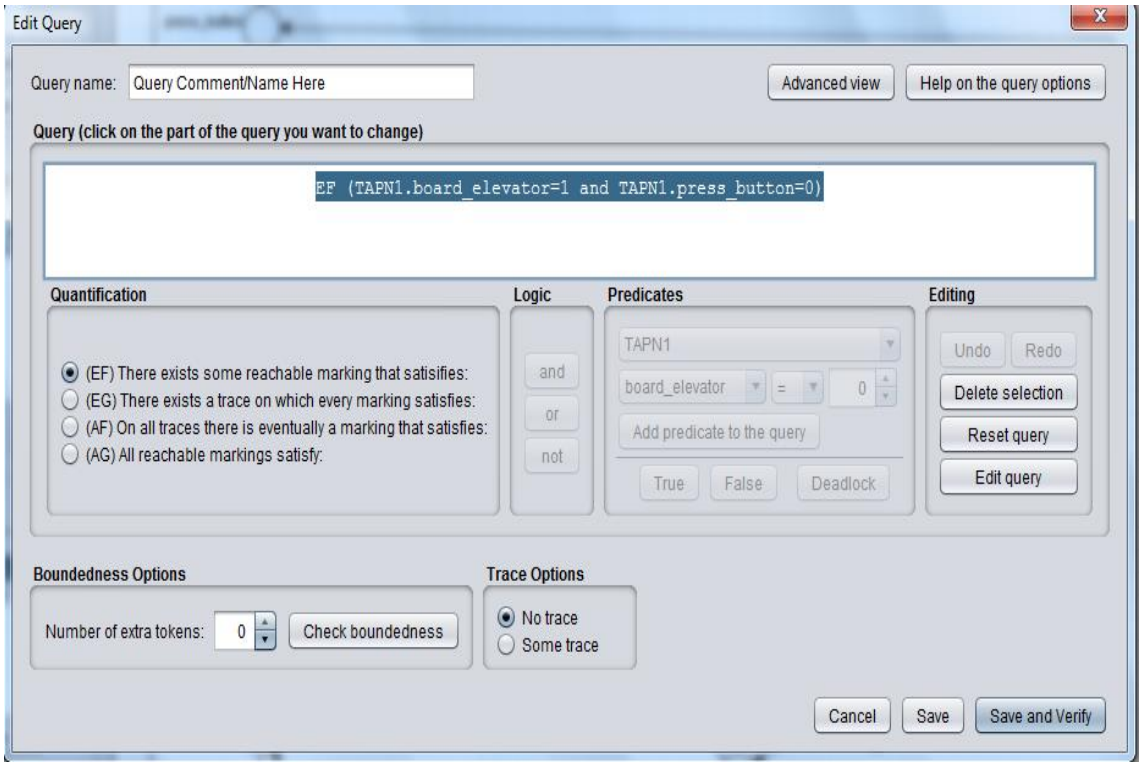

Figure 5.3: Query Editor of TAPAAL tool to verify Elevator Control System

Figure 5.4: Query result of Elevator Control System

Thus, with the formal analysis and model verification it can be observed that the model for Elevator Control System is correct and can be considered for the next phase i.e., system design.

5.3 Summary

The modeling of Elevator control System is illustrated using TAPAAL tool. The model verification is done by wring queries with the help of computation Tree logic (CTL) operators. The queries satisfied all the features of the model. Hence, the model for Elevator Control System is correct.

Chapter 6

Conclusion and Future Work

Analysis of the Tools used

Conclusion

Future work

Chapter 6

Conclusion and Future Work

6.1 Analysis of the Tools used

Advantages and Disadvantages of CPN:

- **Advantages**
  - Provides concept of color.
  - Hierarchical in manner.
  - Provides state space analysis of the model.
- **Disadvantages**
  - performance analysis in the form of transformation to markovian process is not supported.
  - In state space analysis, there is a problem of state explosion.

Advantages and Disadvantages of PIPE:

- **Advantages**
  - Stochastic in nature.
  - Provides GSPN analysis of the model.
- **Disadvantages**
  - only one type of token is supported.

- no hierarchical feature supported.

Advantages and Disadvantages of TAPAAL:

- **Advantages**
  - provides the timed arc concept.
  - use the concept of temporal logic to verify the model.
- **Disadvantages**
  - only one type of token is supported.
  - no hierarchical feature supported.
  - does not provide performance analysis feature.

Advantages of PNML:

- provides the xml based interchangeable format.
- The model generated xml code can be imported by other tools which support PNML format.
- There is no need of modeling the system again and again on different tool for different task.

So, from the above analysis, it is cleared that PNML is the better tool in the field of modeling, analysis and verification of the system as compared to other tools. Since, PNML provides the interchangeable format, it becomes easier to import the designed model of PNML to other tool for further process.

6.2 Conclusion

Embedded system are playing a crucial role in our day to day life. Designing system with parameter related to quality and having high level of complexity is a very difficult task. The model of the system must be unambiguous and portrayed crucial characteristics of the system.

This thesis presents a formal model for embedded system. The performance analysis of the model is carried out using Stochastic Petri nets to analyze the model. The model verification methodology is used to prove whether certain properties, expressed as Computational tree logic formulas, fulfilled by the model.

A brief comparison among various tools used in this thesis is presented and stated that which tool perform better according to the system requirements.

6.3 Future Work

In this thesis, the proposed approach makes an attempt to model, analyze and verify the embedded system using different Petri net tools. The work can be extended to model and evaluate the performance of embedded system using Colored Stochastic Petri nets (CSPN) [25]. CSPN combines the applications of both the Colored Petri nets and Stochastic Petri nets. Thus, this approach can overcome the drawbacks of CPNs and SPNs as Stochastic Petri net does not support type of the tokens and textual representation of the model. Colored Petri net does not support stochasticity of the model. So, CSPN methodology is an approach which can decrease the effort of modeling the system on different Petri net tools to analyze the performance.

Bibliography

[1] R. Kamal, *Embedded systems 2E.* Tata McGraw-Hill Education, 2008.

[2] C. Petri, "Kommunikation mit automaten. bonn: Institut für instrumentelle mathematik, schriften des iim, no. 3 (1962); also, english translation: Communication with automata, griffiss air force base," tech. rep., New York. Tech. Rep. RADC-TR. 1 (suppl. 1), 1966.

[3] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, Apr, 1989.

[4] K. Jensen, "Coloured petri nets," in *Petri nets: central models and their properties*, pp. 248–299, Springer, 1987.

[5] H. Zhang, H. Zhang, M. Gu, and J. Sun, "Modeling a heterogeneous embedded system in coloured petri nets," *Journal of Applied Mathematics*, vol. 2014, 2014.

[6] N. R. Adam, V. Atluri, and W.-K. Huang, "Modeling and analysis of workflows using petri nets," *Journal of Intelligent Information Systems*, vol. 10, no. 2, pp. 131–158, 1998.

[7] W. M. van der Aalst, C. Stahl, and M. Westergaard, "Strategies for modeling complex processes using colored petri nets," in *Transactions on Petri Nets and Other Models of Concurrency VII*, pp. 6–55, Springer, 2013.

[8] O. Baldello, J.-C. Fabre, and M. Roy, "Modeling distributed real-time systems using adaptive petri nets," pp. 7–8, Saint-Malo, France, May, 2011.

[9] F. Bause and P. S. Kritzinger, *Stochastic Petri Nets.* Springer, 2002.

[10] M. K. Molloy, "Performance analysis using stochastic petri nets," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 913–917, Sep,1982.

[11] S. Bernardi and J. Campos, "Computation of performance bounds for real-time systems using time petri nets," *Industrial Informatics, IEEE Transactions on*, vol. 5, no. 2, pp. 168–180, May, 2009.

[12] M. H. Cintra, W. V. Ruggiero, and L. D. S. Integráveis, "A simulation technique for performance analysis of generic petri net models of computer systems1,"

[13] X. He, "Specifying and verifying real-time systems using time petri nets and real-time temporal logic," in *Computer Assurance, 1991. COMPASS'91, Systems Integrity, Software Safety and Process Security. Proceedings of the Sixth Annual Conference on*, pp. 135–140, IEEE, June, 1991.

[14] W. Van Der Aalst, "Challenges in business process management: Verification of business processes using petri nets," *Bulletin of the EATCS*, vol. 80, pp. 174–199, 2003.

[15] H. Boucheneb and R. Hadjidj, "Ctl* model checking for time petri nets," *Theoretical Computer Science*, vol. 353, no. 1, pp. 208–227, 2006.

[16] F. Strobl and A. Wisspeintner, "Specification of an elevator control system–an autofocus case study," *Institutsbericht, Technische Universitaet Muenchen, Institut fuer Informatik*, 1999.

[17] J. Billington, S. Christensen, K. Van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber, "The petri net markup language: concepts, technology, and tools," in *Applications and Theory of Petri Nets 2003*, pp. 483–505, Springer, 2003.

[18] I. Jacobson, G. Booch, and J. Rumbaugh, "The unified software development process–the complete guide to the unified process from the original designers," *Rational Software Corporation, US*, 1999.

[19] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, "Modelling with generalized stochastic petri nets," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 2, p. 2, 1998.

[20] R. N. Hiscott, "Markov chain analysis," *Mathematical Geology*, vol. 14, no. 5, pp. 543–544, 1982.

[21] P. Bonet, C. M. Lladó, R. Puijaner, and W. J. Knottenbelt, "Pipe v2. 5: A petri net tool for performance modelling," in *Proc. 23rd Latin American Conference on Informatics (CLEI 2007)*, 2007.

[22] T. Bolognesi and F. Lucidi, "Timed process algebras with urgent interactions and a unique powerful binary operator," in *Real-Time: Theory in Practice*, pp. 124–148, Springer, Jan, 1992.

[23] J. Byg, K. Y. Jørgensen, and J. Srba, "Tapaal: Editor, simulator and verifier of timed-arc petri nets," in *Automated Technology for Verification and Analysis*, pp. 84–89, Springer, 2009.

[24] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in *Formal methods for the design of real-time systems*, pp. 200–236, Springer, 2004.

[25] Z. Wang, M. Atli, and H. K. Adjallah, "Coloured stochastic petri nets modelling for the reliability and maintenance analysis of multi-state multi-unit systems," *Journal of Manufacturing Technology Management*, vol. 25, no. 4, pp. 476–490, 2014.

Dissemination of Work

Communicated

Prerna kanojia and Santanu Ku. Rath, "Validation of Requirements for Embedded Software using Petri nets" 3^{rd} International Conference on Advances in Computing, Communication and Informatics (ICACCI), Delhi, September 2014.